

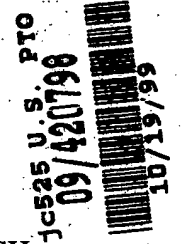
IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In the Patent Application of)

YOSHIHIKO IMAMURA)

Serial No. (Not yet assigned))

Filed: October 19, 1999)

For: PARALLEL PROCESSOR, PARALLEL)
PROCESSING METHOD AND STORING MEDIUM)ATTN:
APPLICATION BRANCHCLAIM TO PRIORITY UNDER 35 U.S.C. 119Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

The benefit of the filing date of the following prior applications filed in the following foreign country is hereby requested and the right of priority provided under 35 U.S.C. 119 is hereby claimed:

Japanese Patent Appl. No. P10-302679, filed October 23, 1998

In support of this claim, filed herewith is a certified copy of said original foreign applications.

Respectfully submitted,

A handwritten signature in black ink, appearing to read 'Ronald P. Kananen', written over a horizontal line.

Ronald P. Kananen
Reg. No. 24,104

Dated: October 19, 1999

RADER, FISHMAN & GRAUER P.L.L.C.
1233 20TH Street, NW
Suite 501
Washington, DC 20036
202-955-3750-Phone
202-955-3751 - Fax

599P11754500

日 本 国 特 許 庁

PATENT OFFICE
JAPANESE GOVERNMENT

15825 U.S. PTO
09/420798
10/19/99

別紙添付の書類に記載されている事項は下記の出願書類に記載されて
いる事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed
with this Office.

出 願 年 月 日
Date of Application:

1998年10月23日

出 願 番 号
Application Number:

平成10年特許願第302679号

出 願 人
Applicant(s):

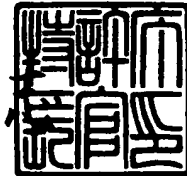
ソニー株式会社

CERTIFIED COPY OF
PRIORITY DOCUMENT

1999年 8月18日

特許庁長官
Commissioner,
Patent Office

伴佐山建



出証番号 出証特平11-3057919

【書類名】 特許願

【整理番号】 9800521607

【提出日】 平成10年10月23日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 5/00

【発明の名称】 並列処理装置、並列処理方法および記録媒体

【請求項の数】 22

【発明者】

【住所又は居所】 東京都品川区北品川6丁目7番35号 ソニー株式会社
内

【氏名】 今村 義彦

【特許出願人】

【識別番号】 000002185

【氏名又は名称】 ソニー株式会社

【代表者】 出井 伸之

【代理人】

【識別番号】 100094053

【弁理士】

【氏名又は名称】 佐藤 隆久

【手数料の表示】

【予納台帳番号】 014890

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9707389

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 並列処理装置、並列処理方法および記録媒体

【特許請求の範囲】

【請求項 1】

プログラムに記述された命令に基づいて相互に並列に処理を行い、相互間で通信可能な複数の処理手段を有する並列処理装置において、

一の前記処理手段は、待機命令を実行したときに前記プログラムに基づいた処理を中断して待機状態になり、他の前記処理手段による待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開し、

前記他の処理手段は、前記待機解除命令を実行後、処理を中断することなく次の命令を実行する

並列処理装置。

【請求項 2】

前記他の処理手段は、同期待ち命令を実行して同期待ち状態になり、前記一の処理手段による前記同期待ち命令に対応する前記待機命令の実行に基づいて前記同期待ち状態を解除する

請求項 1 に記載の並列処理装置。

【請求項 3】

前記プログラムを記憶する第 1 の記憶手段と、

前記複数の処理手段にそれぞれ対応して設けられ、対応する処理手段で実行されるプログラムを前記第 1 の記憶手段から読み込み、当該読み込んだプログラムに記述された命令を前記処理手段に供給し、前記第 1 の記憶手段に比べてアクセス速度が速い第 2 の記憶手段と

をさらに有し、

前記第 2 の記憶手段は、対応する前記処理手段が前記待機状態であるとき、当該待機状態になる前に当該処理手段に供給していた前記プログラムの記憶を継続する

請求項 1 に記載の並列処理装置。

【請求項 4】

前記第 2 の記憶手段は、対応する前記処理手段がプログラムの終了を示すプログラム終了命令を実行するまで、前記プログラムの記憶を継続する
請求項 3 に記載の並列処理装置。

【請求項 5】

前記処理手段は、前記第 1 の記憶手段に記憶されているプログラムを他の前記処理手段で実行させることを指示するプログラム実行指示命令を実行し、

当該他の処理手段に対応する前記第 2 の記憶手段は、前記プログラム実行指示命令によって実行が指示されたプログラムを前記第 1 の記憶手段から読み込んで記憶する

請求項 1 に記載の並列処理装置。

【請求項 6】

前記処理手段による前記プログラム実行指示命令に基づいて、当該プログラム実行指示命令によって実行が指示されたプログラムを実行させる前記処理手段を決定し、当該プログラム実行指示命令によって実行が指示されたプログラムを前記第 1 の記憶手段から前記決定した処理手段に対応する前記第 2 の記憶手段に読み込むプログラム実行割り付け手段

をさらに有する請求項 5 に記載の並列処理装置。

【請求項 7】

前記プログラム実行指示命令によって実行が指示されたプログラムに記述された前記待機命令に基づいて前記他の処理手段が待機状態になったときに、前記プログラム実行指示命令を実行した前記処理手段が前記同期解除命令を実行する

請求項 5 に記載の並列処理装置。

【請求項 8】

前記プログラム実行指示命令によって実行が指示されたプログラムに記述された前記待機命令に基づいて前記他の処理手段が待機状態になったときに、前記プログラム実行指示命令を実行した前記処理手段以外の前記処理手段が前記同期解除命令を実行する

請求項 5 に記載の並列処理装置。

【請求項 9】

前記複数の処理手段と、
当該複数の処理手段を接続する共有バスと
が単体の半導体チップ内に組み込まれている
請求項 1 に記載の並列処理装置。

【請求項 10】

プログラムに記述された命令に基づいて相互に並列に処理を行い、相互間で通信可能な複数の処理手段を有する並列処理装置において、

一の前記処理手段は、待機命令を実行すると前記プログラムに基づいた処理を中断して待機状態になり、他の前記処理手段による待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開し、

前記他の処理手段は、前記待機解除命令を実行したときに、前記一の処理手段が前記待機状態になっていない場合に、前記一の処理手段が前記待機状態になるまで同期待ち状態になる

並列処理装置。

【請求項 11】

前記他の処理手段は、前記待機解除命令を実行したときに、前記一の処理手段が前記待機状態になっている場合に、前記待機解除命令を実行後、処理を中断することなく次の命令を実行する

請求項 10 に記載の並列処理装置。

【請求項 12】

前記プログラムを記憶する第 1 の記憶手段と、

前記複数の処理手段にそれぞれ対応して設けられ、対応する処理手段で実行されるプログラムを前記第 1 の記憶手段から読み込み、当該読み込んだプログラムに記述された命令を前記処理手段に供給し、前記第 1 の記憶手段に比べてアクセス速度が速い第 2 の記憶手段と

をさらに有し、

前記第 2 の記憶手段は、対応する前記処理手段が前記待機状態であるとき、当該待機状態になる前に当該処理手段に供給していた前記プログラムの記憶を継続

する

請求項 10 に記載の並列処理装置。

【請求項 13】

前記第 2 の記憶手段は、対応する前記処理手段がプログラムの終了を示すプログラム終了命令を実行するまで、前記プログラムの記憶を継続する

請求項 12 に記載の並列処理装置。

【請求項 14】

プログラムに記述された命令に基づいて相互に並列に処理を行い、相互間で通信可能な複数の処理手段と、

前記プログラムを記憶する第 1 の記憶手段と、

前記複数の処理手段にそれぞれ対応して設けられ、対応する処理手段で実行されるプログラムを前記第 1 の記憶手段から読み込み、当該読み込んだプログラムに記述された命令を前記処理手段に供給し、前記第 1 の記憶手段に比べてアクセス速度が速い第 2 の記憶手段と

を有する並列処理装置において、

前記処理手段は、待機命令を実行したときに前記プログラムに基づいた処理を中断して待機状態になり、他の前記処理手段による待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開し、

前記第 2 の記憶手段は、対応する前記処理手段が前記待機状態であるとき、当該待機状態になる前に当該処理手段に供給していた前記プログラムの記憶を継続する

並列処理装置。

【請求項 15】

前記第 2 の記憶手段は、対応する前記処理手段がプログラムの終了を示すプログラム終了命令を実行するまで、前記プログラムの記憶を継続する

請求項 14 に記載の並列処理装置。

【請求項 16】

前記処理手段は、前記第 1 の記憶手段に記憶されているプログラムを他の前記処理手段で実行させることを指示するプログラム実行指示命令を実行し、

当該他の処理手段に対応する前記第2の記憶手段は、前記プログラム実行指示命令によって実行が指示されたプログラムを前記第1の記憶手段から読み込んで記憶する

請求項14に記載の並列処理装置。

【請求項17】

前記処理手段による前記プログラム実行指示命令に基づいて、当該プログラム実行指示命令によって実行が指示されたプログラムを実行させる前記処理手段を決定し、当該プログラム実行指示命令によって実行が指示されたプログラムを前記第1の記憶手段から前記決定した処理手段に対応する前記第2の記憶手段に読み込むプログラム実行割り付け手段

をさらに有する請求項16に記載の並列処理装置。

【請求項18】

プログラムに記述された命令に基づいて少なくとも第1の処理および第2の処理を相互に並列に行う並列処理方法において、

前記第1の処理は、待機命令の実行によって前記プログラムに基づいた処理を中断して待機状態になり、第2の処理における待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開し、

前記第2の処理は、前記待機解除命令を実行後、処理を中断することなく次の命令を実行する

並列処理方法。

【請求項19】

前記第2の処理は、同期待ち命令の実行によって同期待ち状態になり、前記第1の処理における前記同期待ち命令に対応する前記待機命令の実行に基づいて前記同期待ち状態を解除する

請求項18に記載の並列処理方法。

【請求項20】

プログラムに記述された命令に基づいて少なくとも第1の処理および第2の処理を相互に並列に行う並列処理方法において、

前記第1の処理は、待機命令の実行によって前記プログラムに基づいた処理を

中断して待機状態になり、第2の処理における待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開し、

前記第2の処理は、前記待機解除命令の実行時に、前記第1の処理が前記待機状態になっていない場合に、前記第1の処理が前記待機状態になるまで同期待ち状態になる

並列処理方法。

【請求項 21】

プログラムに記述された命令に基づいて相互に並列に行われる第1の処理および第2の処理の手順をコンピュータで読み取り可能に記録した記録媒体であって

前記第1の処理は、待機命令の実行によって前記プログラムに基づいた処理を中断して待機状態になり、第2の処理における待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開する処理であり、

前記第2の処理は、前記待機解除命令を実行後、処理を中断することなく次の命令を実行する処理である

記録媒体。

【請求項 22】

プログラムに記述された命令に基づいて相互に並列に行われる第1の処理および第2の処理の手順をコンピュータで読み取り可能に記録した記録媒体であって

前記第1の処理は、待機命令の実行によって前記プログラムに基づいた処理を中断して待機状態になり、第2の処理における待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開する処理であり、

前記第2の処理は、前記待機解除命令の実行時に、前記第1の処理が前記待機状態になっていない場合に、前記第1の処理が前記待機状態になるまで同期待ち状態になる処理である

記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、並列処理装置、並列処理方法および当該方法の手順をコンピュータで読み取り可能に記録した記録媒体に関する。

【0002】

【従来の技術】

例えば、Unix（商標名）などのOS (Operating System) を搭載したシングルプロセッサでは、マルチタスク環境でのプログラムの実行に際してローカルメモリに同時に存在する複数のプログラムの進行を管理する機能が必要であり、当該機能において「プログラム」という言葉を明確な対象概念とするために「プロセス」という概念が用いられている。プロセスは、独立したプログラムの実行単位であり、当該プログラムが独自にアクセスできるメモリ空間（ユーザメモリ空間）をローカルメモリに設定する。また、プログラムの実行はプロセスの起動を意味し、プログラムの終了はプロセスの消滅を意味する。また、プロセスは、他のプロセスを起動および消滅させたり、他のプロセスと通信を行うことができる。

【0003】

ところで、シングルプロセッサでは、CPU (Central Processing Unit) が1個であるため、起動できるプロセスは、任意の時刻で最大1個である。従って、シングルプロセッサでは、独立した複数のプログラムにユーザメモリ空間を同時に割り当て、時分割 (Time Sharing) 方式で当該複数のプログラムを交互に実行して複数のプロセスを交互に起動することで、マルチタスク環境を実現している。

このとき、一のプロセスが起動状態にある場合に、他のプロセスは待機状態になっている。

【0004】

上述したマルチタスク環境下では、複数のプロセス相互間でのメッセージ通信を以下に示すよう行う。

すなわち、シングルプロセッサでは、前述したように、任意の時刻で起動状態

になるプロセスは最大1個であるため、メッセージを送信する一のプロセスが起動状態にあるときには、当該メッセージを受信する他のプロセスは待機状態になっている。従って、メッセージを送信する起動状態のプロセスは、O/Sのカーネル内のプロセス管理タスクを呼び出し、当該メッセージを受信するプロセスが前回待機状態に遷移する直前の起動状態を記憶しているメモリ上のテーブル（通常、スレッドのコンテキストを保存しているテーブル）にメッセージを送信する旨を書き込む。そして、メッセージ受信するプロセスが、次に起動状態に遷移すると、当該テーブルを参照することで、メッセージを受信したことを知り、それに応じた処理を行う。一方、例えば、メッセージを受信するプロセスが当該メッセージの受信を条件に次の処理に進むものであり、起動状態に遷移してテーブルを参照したときにメッセージの受信の事実が無いと判断した場合には、当該プロセスは待機状態になる。そして、当該プロセスは、メッセージの受信を確認した後、起動状態に遷移する。

【0005】

一方、例えば共有バスを介して接続された複数のCPUが相互に独立した複数のプログラムを並列に実行するマルチプロセッサでは、通常、1個のCPUにおいて任意の時刻で最大1個のプロセスが起動状態となり、相互に異なるCPU上で複数のプロセスが同時に起動状態となり得る。

そして、プロセス相互間での通信は、ユーザプログラム（アプリケーションプログラム）内に明示された命令コードに基づいて、例えば、送信側のプロセスが共有バスにメッセージを流し、共有バスを監視しているアービタが受信側のプロセスに当該メッセージを通知して行われる。従って、プロセス相互間でメッセージ通信を実現するには、メッセージの送信側のプロセスと受信側のプロセスとの双方が起動状態である必要がある。

このように、通常、マルチプロセッサでは、前述したシングルプロセッサのようなプロセス管理タスクを用いたメッセージ通信は行われていない。すなわち、マルチプロセッサでは、プロセス管理タスクというものが存在しない。

【0006】

ところで、マルチプロセッサでは、並列に動作する複数のプロセス相互間で同

期をとる必要がある場合に、上述したメッセージ通信を用いて同期を実現している。

以下、マルチプロセッサにおける従来のプロセス相互間の同期方法について説明する。

まず、一般的なマルチプロセッサの構成について説明する。

図5は、一般的なマルチプロセッサ1の構成図である。

図5に示すように、マルチプロセッサ1は、共有バス17を介して例えば4個のプロセッサエレメント11₁～11₄を接続した構成をしている。また、共有バス17には、共有メモリ15およびアービタ16が接続されている。

【0007】

ここで、プロセッサエレメント11₁は、例えば、図6に示すように、プロセッサコア31およびローカルメモリ32を有し、共有バス17を介して共有メモリ15から読み出したユーザプログラムをローカルメモリ32に記憶し、ローカルメモリ32に記憶しているユーザプログラムの命令コードをプロセッサコア31に順次に供給して実行する。プロセッサエレメント11₂～11₄は、例えば、プロセッサエレメント11₁と同じ構成をしている。

【0008】

また、アービタ16は、プロセッサエレメント11₁～11₄の実行状態（処理の負荷など）を監視し、当該実行状態に基づいて、共有メモリ15に記憶されているソフトウェアリソースをハードウェアリソースであるプロセッサエレメント11₁～11₄に割り当てる。具体的には、アービタ16は、共有メモリ15に記憶されているユーザプログラムをプロセッサエレメント11₁～11₄の図6に示すローカルメモリ32に読み込む。

アービタ16は、例えば、図7に示すように、ユーザプログラムであるメインプログラムPr g__AおよびサブプログラムPr g__B, Pr g__C, Pr g__D, Pr g__Eを、同時刻あるいは異なる時刻に、図7中矢印で示すプロセッサエレメント11₁～11₄のローカルメモリ32に読み込む。

【0009】

次に、図5に示すマルチプロセッサ1における従来のプログラム（プロセス）

相互間での同期方法について説明する。

まず、共有メモリ 15 に記憶されているメインプログラム Pr g_A が、アービタ 16 によって、プロセッサエレメント 11_1 のローカルメモリ 32 に読み込まれ、図 8 に示すように、メインプログラム Pr g_A に記述された命令コードがプロセッサエレメント 11_1 で順次に実行される。

そして、プロセッサエレメント 11_1 において、命令コード「 gen (Pr g_B) 」が実行されると、その旨を示すメッセージが共有バス 17 を介してアービタ 16 に通知される。そして、アービタ 16 によって、プロセッサエレメント $11_1 \sim 11_4$ の実行状態に基づいて、共有メモリ 15 に記憶されているサブプログラム Pr g_B が、プロセッサエレメント 11_2 のローカルメモリ 32 に読み込まれ、サブプログラム Pr g_B に記述された命令コードがプロセッサエレメント 11_2 で順次に実行される。

【0010】

その後、プロセッサエレメント 11_1 において、命令コード「 gen (Pr g_C) 」が実行されると、その旨を示すメッセージが共有バス 17 を介してアービタ 16 に通知される。そして、アービタ 16 によって、プロセッサエレメント $11_1 \sim 11_4$ の実行状態に基づいて、共有メモリ 15 に記憶されているサブプログラム Pr g_C が、プロセッサエレメント 11_3 のローカルメモリ 32 に読み込まれ、サブプログラム Pr g_C に記述された命令コードがプロセッサエレメント 11_3 で順次に実行される。

【0011】

その後、プロセッサエレメント 11_1 において、命令コード「 gen (Pr g_D) 」が実行されると、その旨を示すメッセージが共有バス 17 を介してアービタ 16 に通知される。そして、アービタ 16 によって、プロセッサエレメント $11_1 \sim 11_4$ の実行状態に基づいて、共有メモリ 15 に記憶されているサブプログラム Pr g_D が、プロセッサエレメント 11_4 のローカルメモリ 32 に読み込まれ、サブプログラム Pr g_D に記述された命令コードがプロセッサエレメント 11_4 で順次に実行される。

【0012】

その後、プロセッサエレメント11₁において、命令コード「wait (Pr g_D)」が実行されると、プロセッサエレメント11₁の処理が同期待ち状態になる。

【0013】

その後、プロセッサエレメント11₄において、サブプログラムPr g_Dの最後の命令コード「end」が実行されると、サブプログラムPr g_Dの終了を示すメッセージが、例えばアービタ16を介してプロセッサエレメント11₁に通知される。これにより、プロセッサエレメント11₁は、同期待ち状態を解除して、次の命令コードを実行する。

【0014】

その後、プロセッサエレメント11₁において、命令コード「wait (Pr g_C)」が実行されると、プロセッサエレメント11₁の処理が同期待ち状態になる。

【0015】

その後、プロセッサエレメント11₃において、サブプログラムPr g_Cの最後の命令コード「end」が実行されると、サブプログラムPr g_Cの終了を示すメッセージが、例えばアービタ16を介してプロセッサエレメント11₁に通知される。これにより、プロセッサエレメント11₁は、同期待ち状態を解除して、次の命令コードを実行する。

【0016】

その後、プロセッサエレメント11₁において、命令コード「gen (Pr g_E)」が実行されると、その旨を示すメッセージが共有バス17を介してアービタ16に通知される。そして、アービタ16によって、プロセッサエレメント11₁～11₄の実行状態に基づいて、共有メモリ15に記憶されているサブプログラムPr g_Eが、例えばプロセッサエレメント11₄のローカルメモリ32に読み込まれ、サブプログラムPr g_Cに記述された命令コードがプロセッサエレメント11₄で順次実行される。

【0017】

その後、プロセッサエレメント 11₁ において、再び、命令コード「gen (Pr g_D)」が実行されると、その旨を示すメッセージが共有バス 17 を介してアービタ 16 に通知される。そして、アービタ 16 によって、プロセッサエレメント 11₁ ~ 11₄ の実行状態に基づいて、共有メモリ 15 に記憶されているサブプログラム Pr g_D が、プロセッサエレメント 11₃ のローカルメモリ 32 に読み込まれ、サブプログラム Pr g_D に記述された命令コードがプロセッサエレメント 11₃ で順次に実行される。

【0018】

【発明が解決しようとする課題】

上述したように、従来のマルチプロセッサ 1 では、異なるプロセッサエレメント上で実行されるプログラム（プロセス）相互間での同期は、一のプロセッサエレメントによる命令コード「wait」の実行で発生した同期待ち状態を、他のプロセッサエレメントにおけるプログラムの実行終了を示す命令コード「end」の実行に基づいて解除するという単純なものである。

すなわち、一のプログラムに基づいたプロセッサエレメントの同期待ち状態は、他のプロセッサエレメントにおけるプログラムの実行が終了しなければ解除できない。従って、異なるプロセッサエレメント上で実行する異なるプログラム相互間で、例えば、プログラムの途中で記述された命令コード相互間で同期をとるなどの多様な形態の同期を実現することができないという問題がある。

【0019】

また、上述した実施形態では、アービタ 16 は、例えば、プロセッサエレメント 11₁ が図 8 に示すメインプログラム Pr g_A を実行中に、メインプログラム Pr g_A によって将来的にどのサブプログラムの呼び出しが行われるかを知ることができない。

そのため、図 8 に示すように、プロセッサエレメント 11₁ における命令コード「gen (Pr g_D)」の最初の実行時と 2 回目の実行時とで、アービタ 16 によって、サブプログラム Pr g_D が異なるプロセッサエレメント 11₃ , 11₄ に割り当てられてしまう可能性がある。この場合には、サブプログラム P

r g_D が比較的短い時間間隔で実行されるにも係わらず、2 回目の実行時において、共有メモリ 15 からプロセッサエレメント 11₃ にサブプログラム P r g_D を読み込む必要があり、プロセッサエレメント 11₃ の待ち時間を長引かせている。

このような事態は、図 6 に示すローカルメモリ 32 のメモリ容量と読み込みを行うプログラムの容量とが同じオーダーである場合に特に頻繁に発生し、マルチプロセッサ 1 の性能を大幅に低下させる要因となっている。

【0020】

本発明は上述した従来技術の問題点に鑑みてなされ、並列に実行されるプログラム相互間で、多様な形態の同期をとることができる並列処理装置、並列処理方法および当該方法の手順をコンピュータで読み取り可能に記録した記録媒体を提供することを目的とする。

また、本発明は、プロセッサエレメントのローカルメモリと共有メモリとの間のユーザプログラムの入れ替えに伴うプロセッサエレメントの待ち時間を短縮できる並列処理装置を提供することを目的とする。

【0021】

【課題を解決するための手段】

上述した従来技術の問題点を解決し、上述した目的を達成するために、本発明の第 1 の観点の並列処理装置は、プログラムに記述された命令に基づいて相互に並列に処理を行い、相互間で通信可能な複数の処理手段を有する並列処理装置であって、一の前記処理手段は、待機命令を実行したときに前記プログラムに基づいた処理を中断して待機状態になり、他の前記処理手段による待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開し、前記他の処理手段は、前記待機解除命令を実行後、処理を中断することなく次の命令を実行する。

【0022】

本発明の第 1 の観点の並列処理装置では、一の処理手段と他の処理手段との間で、双方ともプログラムの実行途中で、プログラム内の待機命令および待機解除命令を用いて、命令レベルで同期がとられる。すなわち、一方のプログラムの実

行終了を伴うことなく、プログラム相互間で同期をとることができる。

【0023】

また、本発明の第1の観点の並列処理装置は、好ましくは、前記他の処理手段は、同期待ち命令を実行して同期待ち状態になり、前記一の処理手段による前記同期待ち命令に対応する前記待機命令の実行に基づいて前記同期待ち状態を解除する。

これにより、前記他の処理手段における前記待機解除命令の実行より前に前記一の処理手段において待機命令が実行されることが回避される。

【0024】

また、本発明の第2の観点の並列処理装置は、プログラムに記述された命令に基づいて相互に並列に処理を行い、相互間で通信可能な複数の処理手段を有する並列処理装置であって、一の前記処理手段は、待機命令を実行すると前記プログラムに基づいた処理を中断して待機状態になり、他の前記処理手段による待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開し、前記他の処理手段は、前記待機解除命令を実行したときに、前記一の処理手段が前記待機状態になっていない場合に、前記一の処理手段が前記待機状態になるまで同期待ち状態になる。

【0025】

すなわち、本発明の第2の観点の並列処理装置では、前記一の処理手段と前記他の処理手段との間で、プログラム内の待機命令および待機解除命令を用いて、命令レベルで同期がとられる。すなわち、一方のプログラムの実行終了を伴うことなく、プログラム相互間で同期をとることができる。また、前記他の処理手段で処理されるプログラム内に、待機解除命令に対応する同期待ち命令が記述されていなくても、前記一の処理手段が前記待機状態になっていない場合には、前記他の処理手段が前記待機状態になるまで同期待ち状態になる。

【0026】

また、本発明の第3の並列処理装置は、プログラムに記述された命令に基づいて相互に並列に処理を行い、相互間で通信可能な複数の処理手段と、前記プログラムを記憶する第1の記憶手段と、前記複数の処理手段にそれぞれ対応して設け

られ、対応する処理手段で実行されるプログラムを前記第1の記憶手段から読み込み、当該読み込んだプログラムに記述された命令を前記処理手段に供給し、前記第1の記憶手段に比べてアクセス速度が速い第2の記憶手段とを有する並列処理装置であって、前記処理手段は、待機命令を実行したときに前記プログラムに基づいた処理を中断して待機状態になり、他の前記処理手段による待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開し、前記第2の記憶手段は、対応する前記処理手段が前記待機状態であるとき、当該待機状態になる前に当該処理手段に供給していた前記プログラムを継続して記憶する。

【0027】

本発明の第3の観点の並列処理装置では、一の前記処理手段が、待機命令を実行したときに前記プログラムに基づいた処理を中断して待機状態になり、他の前記処理手段による待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開する場合に、当該一の処理手段に対応する第2の記憶手段には、前記一の処理手段に供給していたプログラムが継続して記憶される。すなわち、当該プログラムの実行を再開する際に、当該プログラムを再び前記第1の記憶手段から前記第2の記憶手段に読み込む必要がない。

【0028】

また、本発明の第1の観点の並列処理方法は、プログラムに記述された命令に基づいて少なくとも第1の処理および第2の処理を相互に並列に行う並列処理方法であって、前記第1の処理は、待機命令の実行によって前記プログラムに基づいた処理を中断して待機状態になり、第2の処理における待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開し、前記第2の処理は、前記待機解除命令を実行後、処理を中断することなく次の命令を実行する。

【0029】

また、本発明の第2の観点の並列処理方法は、プログラムに記述された命令に基づいて少なくとも第1の処理および第2の処理を相互に並列に行う並列処理方法であって、前記第1の処理は、待機命令の実行によって前記プログラムに基づ

いた処理を中断して待機状態になり、第2の処理における待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開し、前記第2の処理は、前記待機解除命令の実行時に、前記第1の処理が前記待機状態になっていない場合に、前記第1の処理が前記待機状態になるまで同期待ち状態になる。

【0030】

また、本発明の第1の観点の記録媒体は、プログラムに記述された命令に基づいて相互に並列に行われる第1の処理および第2の処理の手順をコンピュータで読み取り可能に記録した記録媒体であって、前記第1の処理は、待機命令の実行によって前記プログラムに基づいた処理を中断して待機状態になり、第2の処理における待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開する処理であり、前記第2の処理は、前記待機解除命令を実行後、処理を中断することなく次の命令を実行する処理である。

【0031】

さらに、本発明の第2の観点の記録媒体は、プログラムに記述された命令に基づいて相互に並列に行われる第1の処理および第2の処理の手順をコンピュータで読み取り可能に記録した記録媒体であって、前記第1の処理は、待機命令の実行によって前記プログラムに基づいた処理を中断して待機状態になり、第2の処理における待機解除命令の実行に基づいて前記待機状態を解除して前記プログラムに基づいた処理を再開する処理であり、前記第2の処理は、前記待機解除命令の実行時に、前記第1の処理が前記待機状態になっていない場合に、前記第1の処理が前記待機状態になるまで同期待ち状態になる処理である。

【0032】

【発明の実施の形態】

以下、本発明の実施形態に係わるマルチプロセッサについて説明する。

第1実施形態

図1は、本実施形態のマルチプロセッサ51の構成図である。

図1に示すように、マルチプロセッサ51は、例えば、共有バス17、処理手段および第2の記憶手段としてのプロセッサエレメント61₁～61₄、第1の

記憶手段としての共有メモリ 65 およびプログラム割り付け手段としてのアービタ 66 を有する。

マルチプロセッサ 51 は、例えば、共有バス 17 を介して、プロセッサエレメント $61_1 \sim 61_4$ 、共有メモリ 65 およびアービタ 66 を相互に接続したバス結合のアーキテクチャを採用しており、これらの構成要素が単体の半導体チップ内に組み込まれている。

【0033】

〔構成要素の概要〕

まず、マルチプロセッサ 51 と前述した図 5 に示すマルチプロセッサ 1 との共通点および相違点について説明する。

共有バス 17 は、前述した図 5 に示す従来のマルチプロセッサ 1 の共有バス 17 と同じである。

また、プロセッサエレメント $61_1 \sim 61_4$ は、前述した図 5 に示すプロセッサエレメント $11_1 \sim 11_4$ と比べて、図 6 に示すようにプロセッサコア 31 およびローカルメモリ 32 を有するというハードウェア構成は同じであるが、前述した従来のマルチプロセッサ 1 では用意されていないプログラム（プロセス）相互間の同期確立のための新たな命令コードを実行するため、当該命令コードの実行に伴う動作が異なる。

また、共有メモリ 65 は、図 5 に示す共有メモリ 15 と比べて、ハードウェア構成は同じであるが、記憶しているユーザプログラムの内容が異なる。

さらに、アービタ 56 は、プロセッサエレメント $51_1 \sim 51_4$ の実行状態（処理の負荷など）を監視し、当該実行状態に基づいて、共有メモリ 55 に記憶されているソフトウェアリソースをハードウェアリソースであるプロセッサエレメント $51_1 \sim 51_4$ に割り当てる点は、図 5 に示すアービタ 16 と同様である。但し、アービタ 56 は、プロセッサエレメント $61_1 \sim 61_4$ において前述した新たな命令コードが実行されたときに、図 5 に示すアービタ 16 とは異なる種々の同期確立のための処理およびプロセッサエレメント $61_1 \sim 61_4$ へのユーザプログラムの割り当て処理を行う。

【0034】

〔命令コード〕

マルチプロセッサ51は、以下に示す命令コードを採用している。

具体的には、マルチプロセッサ51は、命令コードとして、プログラム実行指示命令としての「gen（ユーザプログラム名）」、同期待ち命令としての「wait（ユーザプログラム名）」、待機解除命令としての「cont（ユーザプログラム名）」、待機命令としての「sleep」およびプログラム終了命令としての「end」を有する。なお、マルチプロセッサ51は、その他にも、例えば、汎用のマルチプロセッサにおいて用意している種々の命令コードを有する。

ここで、命令コード「gen（ユーザプログラム名）」、「wait（ユーザプログラム名）」および「end」は前述した図5に示すマルチプロセッサ1が採用している同一名の命令コードと同じであり、命令コード「cont（ユーザプログラム名）」および「sleep」はマルチプロセッサ51が独自に採用している命令コードである。

なお、命令コード「gen（ユーザプログラム名）」、「wait（ユーザプログラム名）」および「cont（ユーザプログラム名）」は、ユーザプログラム名を引数としている。なお、引数の数は任意である。

【0035】

命令コード「gen（ユーザプログラム名）」は、他のプロセッサエレメント51₁～51₄に、引数「ユーザプログラム名」で指定したユーザプログラムの実行を開始することを指示する命令コードである。

命令コード「wait（ユーザプログラム名）」は、引数「ユーザプログラム名」で指定したユーザプログラムが、命令コード「sleep」あるいは「end」を実行するまで、同期待ちを行うことを指示する命令コードである。

命令コード「cont（ユーザプログラム名）」は、他のプロセッサエレメント51₁～51₄で実行されている引数「ユーザプログラム名」で指定されたユーザプログラムが待機状態になっているときに、当該待機状態を解除することを指示する命令コードである。

命令コード「sleep」は、ユーザプログラムの実行を一時的に停止して待

機状態となる命令コードである。

命令コード「end」は、ユーザプログラムの実行を終了する命令コードである。

【0036】

なお、上述した命令コードは、プログラマがユーザプログラム作成時に、ユーザプログラム内に明示して記述してもよいし、コンパイラが必要に応じて自動的に挿入してもよい。

【0037】

〔各構成要素の詳細〕

プロセッサエレメント61₁は、図5に示すように、処理手段としてのプロセッサコア31および第2の記憶手段としてのローカルメモリ32を有する。

なお、本実施形態では、プロセッサエレメント61₁～61₄が相互に同じ構成である場合を例示するが、本発明では、プロセッサエレメント61₁～61₄は同じ構成をしている必要はなく、例えば、プロセッサコア31の実行速度やローカルメモリ32の記憶容量などが異なってもよい。

【0038】

ローカルメモリ32は、共有バス17を介して共有メモリ15から読み出されたユーザプログラムを記憶する。

【0039】

プロセッサコア31は、ローカルメモリ32に記憶されているユーザプログラムの命令コードを順次読み出して実行する。

また、プロセッサコア31は、命令コード「gen（ユーザプログラム名）」を実行すると、引数「ユーザプログラム名」によって特定されるユーザプログラムの実行指示を、図1に示す共有バス17を介してアービタ66に出力する。

また、プロセッサコア31は、命令コード「wait（ユーザプログラム名）」を実行すると、同期待ち状態となり、図1に共有バス17を介してアービタ66から、引数「ユーザプログラム名」によって特定されたユーザプログラムの命令コード「end」あるいは「sleep」が実行されたことを示す通知を入力すると、同期待ち状態を解除して次の命令コードを実行する。

【0040】

また、プロセッサコア31は、命令コード「cont (ユーザプログラム名)」を実行すると、引数「ユーザプログラム名」によって特定されるユーザプログラムの実行の待機状態を解除する指示を共有バス17を介してアービタ66に出力し、処理を中断することなく次の命令コードを実行する。

また、プロセッサコア31は、命令コード「sleep」を実行すると、当該命令コードを実行したことを共有バス17を介してアービタ66に通知すると共に、待機状態となり、その後、共有バス17を介してアービタ66から待機状態を解除する指示を入力したときに待機状態を解除して次の命令コードを実行する。

また、プロセッサコア31は、命令コード「end」を実行すると、当該命令コードを実行したことを共有バス17を介してアービタ66に通知すると共に、プログラムの実行を終了する。

【0041】

共有メモリ65は、前述したような命令コード「gen (ユーザプログラム名)」、「wait (ユーザプログラム名)」、「cont (ユーザプログラム名)」、「sleep」および「end」を含む種々の命令コードを記述した例えばユーザプログラムPrg__a, Prg__b, Prg__c, Prg__d, Prg__eを記憶している。

【0042】

アービタ66は、共有バス17を介してプロセッサエレメント61₁～61₄から、ユーザプログラムの実行指示を入力すると、当該実行指示によって特定されるユーザプログラムを、共有メモリ65から、プロセッサエレメント61₁～61₄の実行状態に基づいて例えば負荷が最も小さいプロセッサエレメント61₁～61₄のローカルメモリ32に読み込む。

また、アービタ66は、ユーザプログラムの実行の待機状態を解除する指示を入力すると、当該指示によって特定されるユーザプログラムを実行しているプロセッサエレメント61₁～61₄に、共有バス17を介して、待機状態の解除を示す指示を出力する。

【0043】

また、アービタ66は、命令コード「sleep」および「end」を実行したことを示す通知をプロセッサエレメント61₁～61₄から入力すると、当該命令コードを含むプログラムの実行指示を出力したプロセッサエレメント61₁～61₄に、当該通知を入力したことを通知する。

【0044】

以下、図1に示すマルチプロセッサ51のプロセッサエレメント61₁～61₄におけるユーザプログラムの実行過程を追いながら、マルチプロセッサ51の動作を説明する。

ここでは、図2に示すように、プロセッサエレメント61₁、61₂、61₃、61₄において、ユーザプログラムPrg__a、Prg__b、Prg__c、Prg__d、Prg__eが実行される場合を例示する。

なお、ユーザプログラムPrg__a、Prg__b、Prg__c、Prg__d、Prg__eは、例えば、磁気ディスク、磁気テープ、光ディスクおよび光磁気ディスクなどのコンピュータによって読み取り可能な記録媒体から図1に示す共有バス17に読み込まれている。

【0045】

まず、アービタ66によって、共有メモリ65からプロセッサエレメント61₁のローカルメモリ32に、図2に示すユーザプログラムPrg__aが読み込まれる。

そして、プロセッサエレメント61₁のプロセッサコア31において、ユーザプログラムPrg__aに記述された命令コードが順次実行される。

具体的には、プロセッサエレメント61₁のプロセッサコア31において、まず、命令コード「gen(Prg__b)」が実行され、当該命令コードの引数で特定されるユーザプログラムPrg__bの実行指示が、図1に示す共有バス17を介してアービタ66に出力される。

そして、アービタ66によって、共有メモリ65から共有バス17を介してプロセッサエレメント61₂のローカルメモリ32に、ユーザプログラムPrg__bが読み込まれる。その後、プロセッサエレメント61₂において、ローカルメ

メモリ32に記憶されたユーザプログラムPrg_bに記述された命令コードが順次に読み出されて実行される。

【0046】

次に、プロセッサエレメント61₁のプロセッサコア31において、命令コード「gen(Prg_c)」および「gen(Prg_d)」が順次に実行され、前述した命令コード「gen(Prg_a)」の場合と同様の処理を経て、プロセッサエレメント61₃および61₄のプロセッサコア31において、ユーザプログラムPrg_cおよびPrg_dの実行がそれぞれ開始する。

【0047】

次に、プロセッサエレメント61₁のプロセッサコア31において、命令コード「wait(Prg_d)」が実行され、同期待ち状態となる。

【0048】

その後、プロセッサエレメント61₄において、ユーザプログラムPrg_dに記述された命令コード「sleep」が実行され、プロセッサエレメント61₄が待機状態になる。

また、プロセッサエレメント61₄において命令コード「sleep」が実行されたことが、共有バス17を介してアービタ66に通知される。そして、アービタ66に当該通知が入力されると、ユーザプログラムPrg_dの実行指示を出力したプロセッサエレメント61₁に、当該通知を入力したことがアービタ66から通知される。

そして、プロセッサエレメント61₁がアービタ66から当該通知を入力すると、同期待ち状態が解除され、プロセッサエレメント61₁において次の命令コードが実行される。

【0049】

次に、プロセッサエレメント61₁のプロセッサコア31において、命令コード「wait(Prg_c)」が実行され、同期待ち状態となる。

【0050】

その後、プロセッサエレメント61₃において、ユーザプログラムPrg_cの最後に記述された命令コード「end」が実行され、プロセッサエレメント6

1₃ におけるユーザプログラム Prg_c の実行が終了する。

また、プロセッサエレメント 61₃ において命令コード「end」が実行されたことが、共有バス 17 を介してアービタ 66 に通知される。

そして、アービタ 66 に当該通知が入力されると、ユーザプログラム Prg_c の実行指示を出力したプロセッサエレメント 61₁ に、当該通知を入力したことがアービタ 66 から通知される。

そして、プロセッサエレメント 61₁ がアービタ 66 から当該通知を入力すると、同期待ち状態が解除され、プロセッサエレメント 61₁ において次の命令コードが実行される。

なお、アービタ 66 は、前述したように、プロセッサエレメント 61₃ において命令コード「end」が実行されたことを示す通知を入力すると、その後、プロセッサエレメント 61₃ の負荷はなくなったと判断し、プロセッサエレメント 61₃ のローカルメモリ 32 を開放する。

【0051】

次に、プロセッサエレメント 61₁ のプロセッサコア 31 において、命令コード「gen(Prg_e)」が実行され、前述した命令コード「gen(Prg_a)」の場合と同様の処理を経て、前述したように開放されたプロセッサエレメント 61₃ のローカルメモリ 32 にユーザプログラム Prg_e が読み込まれ、プロセッサエレメント 61₃ のプロセッサコア 31 においてユーザプログラム Prg_e が実行される。

【0052】

次に、プロセッサエレメント 61₁ において、命令コード「cont(Prg_d)」が実行され、ユーザプログラム Prg_d の実行の待機状態を解除する指示が、共有バス 17 を介してアービタ 66 に出力される。そして、アービタ 66 によって、ユーザプログラム Prg_d を実行しているプロセッサエレメント 61₄ に、共有バス 17 を介して、待機状態の解除を示す指示が出力される。

そして、プロセッサエレメント 61₄ がアービタ 66 から当該指示を入力すると、待機状態が解除され、プロセッサエレメント 61₄ において次の命令コードが実行される。

なお、プロセッサエレメント 61₁ では、命令コード「cont (Pr g__d)」の実行後に、処理を中断することなく、次の命令コードが実行される。

【0053】

次に、プロセッサエレメント 61₁ のプロセッサコア 31において、命令コード「wait (Pr g__d)」が実行され、同期待ち状態となる。

【0054】

その後、プロセッサエレメント 61₄ において、ユーザプログラム Pr g__d の最後に記述された命令コード「end」が実行され、プロセッサエレメント 61₄ におけるユーザプログラム Pr g__d の実行が終了する。

また、プロセッサエレメント 61₄ において命令コード「end」が実行されたことが、共有バス 17を介してアービタ 66に通知される。そして、アービタ 66に当該通知が入力されると、ユーザプログラム Pr g__d の実行指示を出力したプロセッサエレメント 61₁ に、当該通知を入力したことがアービタ 66から通知される。

そして、プロセッサエレメント 61₁ がアービタ 66から当該通知を入力すると、同期待ち状態が解除され、プロセッサエレメント 61₁ において次の命令コードが実行される。

なお、アービタ 66は、前述したように、プロセッサエレメント 61₄ において命令コード「end」が実行されたことを示す通知を入力すると、その後、プロセッサエレメント 61₄ の負荷はなくなったと判断し、プロセッサエレメント 61₄ のローカルメモリ 32を開放する。

【0055】

次に、プロセッサエレメント 61₁ のプロセッサコア 31において、命令コード「wait (Pr g__e)」が実行され、同期待ち状態となる。

【0056】

その後、プロセッサエレメント 61₃ において、ユーザプログラム Pr g__e の最後に記述された命令コード「end」が実行され、プロセッサエレメント 61₃ におけるユーザプログラム Pr g__e の実行が終了する。

また、プロセッサエレメント 61₃ において命令コード「end」が実行され

たことが、共有バス17を介してアービタ66に通知される。そして、アービタ66に当該通知が入力されると、ユーザプログラムPrg__eの実行指示を出力したプロセッサエレメント61₁に、当該通知を入力したことがアービタ66から通知される。

そして、プロセッサエレメント61₁がアービタ66から当該通知を入力すると、同期待ち状態が解除され、プロセッサエレメント61₁において次の命令コードが実行される。

なお、アービタ66は、前述したように、プロセッサエレメント61₃において命令コード「end」が実行されたことを示す通知を入力すると、その後、プロセッサエレメント61₃の負荷はなくなったと判断し、プロセッサエレメント61₃のローカルメモリ32を開放する。

【0057】

以上説明したように、マルチプロセッサ51によれば、プログラムの終了を示す命令コード「end」の他に、プログラムの実行の待機状態を指示する命令コード「sleep」と、当該待機状態を解除する命令コード「cont」とを用いることで、異なるプロセッサエレメント61₁～61₄上で実行されているプログラムの命令コード相互間で同期をとることができる。そのため、マルチプロセッサ51によれば、命令コード相互間で同期とる記述を含むプログラムに基づいた多様な処理を行うことが可能になる。

すなわち、従来のマルチプロセッサ1のように、命令コード「end」によるユーザプログラムの実行終了を伴わずに、ユーザプログラム相互間で同期をとることが可能になる。

【0058】

また、マルチプロセッサ51によれば、図2に示すように、ユーザプログラムPrg__aにおいて、命令コード「cont(Prg__d)」の記述の前に、命令コード「wait(Prg__d)」が記述されているため、ユーザプログラムPrg__dの命令コード「sleep」の実行前に、ユーザプログラムPrg__aの命令コード「cont(Prg__d)」が実行されてしまうことを回避できる。これにより、命令コード「sleep」によるプロセッサエレメント61₄

の待機状態が、プロセッサエレメント61₁における命令コード「cont.(Pr g__d)」の実行によって確実に解除される。

【0059】

また、マルチプロセッサ51によれば、命令コード「sleep」が実行された場合に、当該命令コード「sleep」を実行したことを示す通知をアービタ66が入力することで、当該命令コード「sleep」を実行したプロセッサエレメント61₁～61₄において当該命令コード「sleep」を含むユーザプログラムの実行が将来再び再開されることをアービタ66が知ることができる。そのため、アービタ66によって、当該ユーザプログラムが他のユーザプログラムと入れ替えられてしまうことを回避することが可能になり、ユーザプログラムの読み込み動作の回数を削減して処理時間の短縮が図れる。

具体的には、図2に示す例において、共有メモリ65からプロセッサエレメント61₄のローカルメモリ32に、ユーザプログラムPr g__dを1回のみ読み込めばよく、ユーザプログラムPr g__dの読み込みによるプロセッサエレメント61₄の待ち時間を短くできる。また、ユーザプログラムPr g__dの実行を再開するときには、プロセッサエレメント61₁における命令コード「cont (Pr g__d)」の実行によって、その旨がプロセッサエレメント61₄に瞬時に通知されることから、高速な応答が要求されるリアルタイム性が問われるユーザプログラムを実行する場合には特に有効である。

すなわち、図8を用いて前述したような、プロセッサエレメント11₁における命令コード「gen (Pr g__D)」の2回目の実行時に、ユーザプログラムPr g__Dを再びローカルメモリ32に読み込むという無駄な動作の発生を回避できる。

【0060】

上述したような共有メモリ65からローカルメモリ32へのユーザプログラムの無駄な読み込みを回避することによる効果は、ローカルメモリ32のメモリ容量と読み込みを行うユーザプログラムの容量とが同じオーダーである場合に特に顕著である。

【0061】

第2実施形態

本実施形態のマルチプロセッサは、基本的には、前述した第1実施形態のマルチプロセッサ51と同じであるが、上述した命令コード「cont」の代わりに、待機解除命令として以下に示す命令コード「cont_a」を用いる点がマルチプロセッサ51とは異なる。

すなわち、前述した第1実施形態の命令コード「cont」は、一のプロセッサエレメントにおける命令コード「cont」の実行が、他のプロセッサエレメントにおける当該命令コード「cont」に対応する命令コード「sleep」の実行よりも遅いと当該命令コード「sleep」による当該他のプロセッサエレメントの待機状態を解除できない。このような事態を回避するために、第1実施形態では、例えば、図2に示すように、ユーザプログラムPr g_aには、命令コード「cont (Pr g_d)」の記述より前に、ユーザプログラムPr g_dの命令コード「sleep」の実行によって同期待ちが解除される命令コード「wait (Pr g_d)」を記述する必要があった。

【0062】

本実施形態では、第1実施形態の命令コード「cont」の代わりに、新たな命令コード「cont_a」を用いて、それ以前に命令コード「wait」を記述することを不要にしている。

命令コード「cont_a」は、命令コード「cont」と同様に、引数としてユーザプログラム名を指定する。すなわち、命令フォーマットは「cont_a (ユーザプログラム名)」となる。

また、図1に示すプロセッサエレメント61₁～61₄のプロセッサコア31は、命令コード「cont_a (ユーザプログラム名)」を実行すると、引数「ユーザプログラム名」によって特定されるユーザプログラムの実行の待機状態を解除する指示を図1に示す共有バス17を介してアービタ66に出力する。

プロセッサコア31は、命令コード「cont_a (ユーザプログラム名)」を実行すると、その後、処理を中断することなく、次の命令コードを実行する。

また、プロセッサエレメント61₁～61₄は、アービタ66から逆同期待ち

指示を入力すると、その後、アービタ 66 から逆同期待ち状態を解除する指示を入力するまで逆同期待ち状態となる。

【0063】

アービタ 66 は、ユーザプログラムの実行の待機状態を解除する指示を入力すると、当該ユーザプログラムが待機状態である否かを判断し、待機状態であると判断した場合に、当該待機状態のユーザプログラムが割り当てられたプロセッサエレメント 61₁ ~ 61₄ に、共有バス 17 を介して、待機状態の解除を示す指示を出力する。

一方、アービタ 66 は、上記判断において、当該ユーザプログラムが待機状態ではないと判断した場合には、ユーザプログラムの実行の待機状態を解除する指示を出力したプロセッサエレメント 61₁ ~ 61₄ に逆同期待ち指示を出力する。その後、アービタ 66 は、当該ユーザプログラムから、命令コード「sleep」を実行したことの通知を受けると、上記逆同期待ち状態となっているプロセッサエレメント 61₁ ~ 61₄ に、逆同期待ち解除を示す指示を出力すると共に、当該命令コード「sleep」を実行したプロセッサエレメント 61₁ ~ 61₄ に、共有バス 17 を介して待機状態を解除する指示を出力する。

【0064】

以下、図 1 に示すプロセッサエレメント 61₁ ~ 61₄ におけるユーザプログラムの実行過程を追いながら、本実施形態のマルチプロセッサの動作を説明する。

ここでは、図 3 に示すように、図 1 に示すプロセッサエレメント 61₁, 61₂, 61₃, 61₄ において、ユーザプログラム Prg__a a, Prg__b, Prg__c, Prg__d, Prg__e が実行される場合を例示する。

図 3 に示すユーザプログラム Prg__a a は、図 2 に示す命令コード「cont (Prg__d)」およびその前の命令コード「wait (Prg__d)」の代わりに命令コード「cont__a (Prg__d)」を記述している点が前述したユーザプログラム Prg__a とは異なる。また、図 3 に示すユーザプログラム Prg__b, Prg__c, Prg__d, Prg__e は、それぞれ図 2 に示すユーザプログラム Prg__b, Prg__c, Prg__d, Prg__e と同じである。

【0065】

先ず、図3に示すように、プロセッサエレメント61₁において、ユーザプログラムPr g__a aに記述された命令コード「gen (Pr g__b)」、
「gen (Pr g__c)」、
「gen (Pr g__d)」、
「wait (Pr g__c)」
が順に実行される。

これらの処理は、図2を用いて前述した同一名の命令コードの処理と同じである。

【0066】

そして、命令コード「wait (Pr g__c)」による同期待ち状態が解除されると、プロセッサエレメント61₁のプロセッサコア31において、命令コード「cont__a (Pr g__d)」が実行される。

これにより、ユーザプログラムPr g__dの実行の待機状態を解除する指示が、図1に示す共有バス17を介してプロセッサエレメント61₁からアービタ66に出力される。

そして、アービタ66において、プロセッサエレメント61₄で実行されているユーザプログラムPr g__dが待機状態である否かが判断され、この場合には、待機状態となっていないので、プロセッサエレメント61₁に逆同期待ち指示が出力される。

そして、当該逆同期待ち指示を入力したプロセッサエレメント61₁は、逆同期待ち状態となる。

その後、プロセッサエレメント61₄において、ユーザプログラムPr g__dの命令コード「sleep」が実行され、当該命令コード「sleep」を実行したことを示す通知が、共有バス17を介してプロセッサエレメント61₄からアービタ66に出力される。そして、当該通知に基づいて、共有バス17を介してアービタ66からプロセッサエレメント61₁に、逆同期待ち状態を解除する指示が出力され、プロセッサエレメント61₁の逆同期待ちが解除される。

次に、プロセッサエレメント61₁において、ユーザプログラムPr g__a aの命令コード「gen (Pr g__e)」、
「wait (Pr g__d)」、
「wait (Pr g__e)」が実行される。当該実行による処理は、図2を用いて前述

した場合と同じである。

【0067】

なお、例えば、図4に示すように、プロセッサエレメント61₄における命令コード「sleep」が、プロセッサエレメント61₁における「cont__a (Pr g__d)」の実行より早いタイミングで行われた場合には、プロセッサエレメント61₄は、命令コード「sleep」を実行してから待機状態となり、プロセッサエレメント61₁における命令コード「cont__a (Pr g__d)」の実行に基づいた待機解除指示をアービタ66から入力したときに、当該待機状態を解除する。

【0068】

以上説明したように、本実施形態のマルチプロセッサによれば、上述したような命令コード「cont__a」を用いることで、例えば、図2に示す第1実施形態のように、命令コード「cont」の記述より前に命令コード「sleep」の実行によって同期待ちが解除される命令コード「wait」を記述する必要がない。すなわち、例えば、図2に示す場合に、ユーザプログラムPr g__dの命令コード「sleep」の実行が、命令コード「cont__a (Pr g__d)」より前のタイミングで実行された場合でも、プロセッサエレメント61₁が逆同期待ちになり、ユーザプログラムPr g__a aとPr g__dとの間の同期が保証される。

そのため、プログラマはユーザプログラムPr g__dの命令コード「sleep」の実行タイミングを意識せずにユーザプログラムPr g__a aを記述でき、プログラマの負担を軽減できる。

【0069】

本発明は上述した実施形態には限定されない。

例えば、上述した実施形態では、プロセッサエレメント61₁～61₂が、命令コード「cont」および「cont__a」の実行に応じて出力した待機状態を解除する指示を、アービタ66を介して、待機状態のユーザプログラムを実行しているプロセッサエレメント61₁～61₄に出力する場合を例示したが、待機状態のユーザプログラムを実行しているプロセッサエレメント61₁～61₄

が共有バス 17 を監視し、待機状態を解除する指示をアービタ 66 を介さずに入力するようにしてもよい。

また、同様に、プロセッサエレメント 61₁ ~ 61₄ から出力された命令コード「sleep」および「end」を実行したことを示す通知を、アービタ 66 を介さずに、共有バス 17 を監視している対応するプロセッサエレメント 61₁ ~ 61₄ が直接入力するようにしてもよい。

【0070】

また、上述した実施形態では、ユーザプログラム Prg_a, Prg_aa のみに、命令コード「gen」, 「cont」および「cont_a」が記述されている場合を例示したが、これらの命令コードはユーザプログラム Prg_b, Prg_c, Prg_d にも記述されていてもよい。

また、複数のユーザプログラムに、命令コード「sleep」が記述されていてもよい。

【0071】

また、上述した実施形態では、4 個の同一構成のプロセッサエレメントを有するマルチプロセッサを例示したが、プロセッサエレメントの数は 2 以上であれば任意であり、複数のプロセッサエレメントの構成が異なってもよい。

【0072】

また、上述した実施形態では、図 1 に示す構成要素が同一半導体チップ内に組み込まれている場合を例示したが、本発明は、例えば、図 1 に示すプロセッサエレメント 61₁ ~ 61₄ が、ネットワークを介して接続された相互に異なるコンピュータ内にそれぞれ組み込まれた分散処理システムにも適用できる。

【0073】

【発明の効果】

以上説明したように、本発明の並列処理装置によれば、複数の処理手段で並列に実行されるプログラム相互間で、多様な形態の同期をとることができる。

また、本発明の並列処理方法および記録媒体によれば、並列に行われる複数の処理の相互間で、多様な形態の同期をとることができる。

また、本発明の並列処理装置によれば、処理手段が待機命令を実行することで

、当該処理手段に対応する第2の記憶手段への第1の記憶手段からのプログラムの読み込みを制御でき、第1の記憶手段から第2の記憶手段へのプログラムの読み込み動作の回数を削減し、処理手段の待ち時間を短縮できる。

【図面の簡単な説明】

【図1】

図1は、本発明の第1実施形態のマルチプロセッサの構成図である。

【図2】

図2は、図1に示す本発明の第1実施形態のマルチプロセッサの動作を説明するための図である。

【図3】

図3は、本発明の第2実施形態のマルチプロセッサの動作を説明するための図である。

【図4】

図3は、本発明の第2実施形態のマルチプロセッサのその他の動作を説明するための図である。

【図5】

図5は、一般的なマルチプロセッサの構成図である。

【図6】

図6は、図1および図5に示すプロセッサエレメントの内部構成図である。

【図7】

図7は、図5に示すマルチプロセッサの各プロセッサエレメントへのプログラムの割り付けを説明するための図である。

【図8】

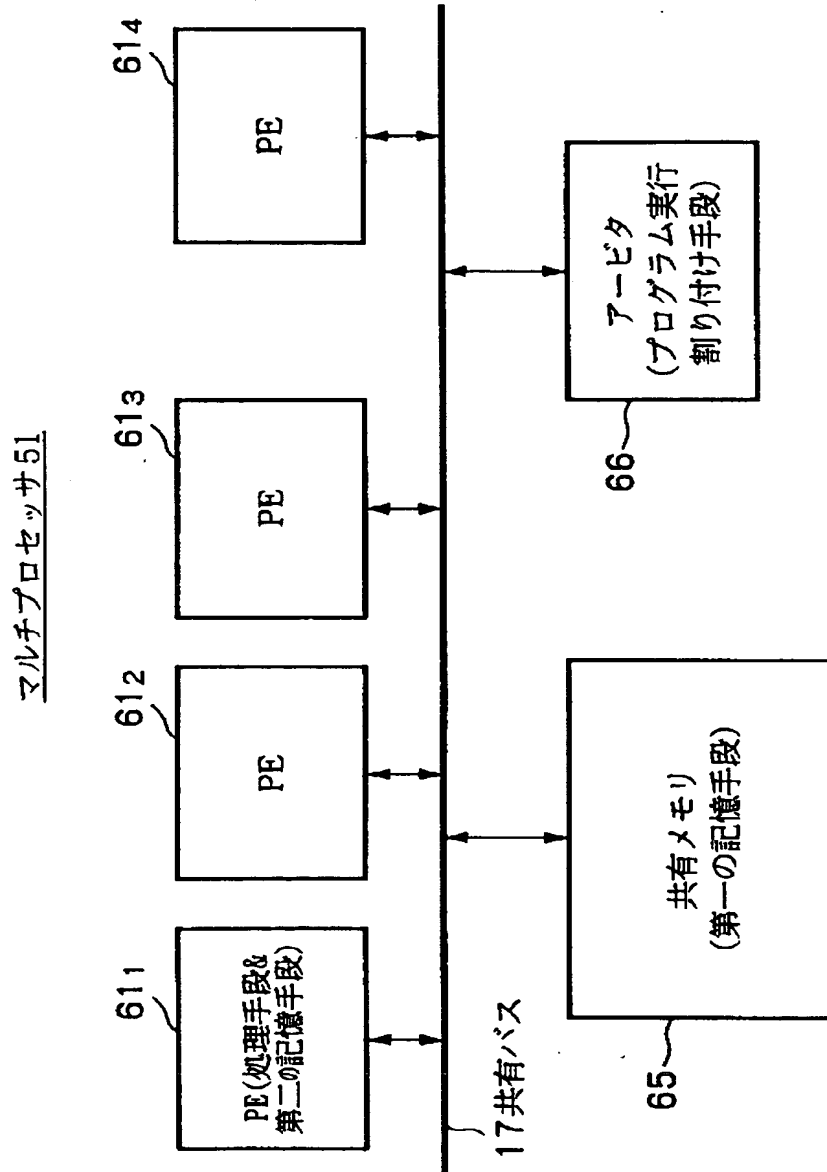
図8は、図5に示す一般的なマルチプロセッサの動作を説明するための図である。

【符号の説明】

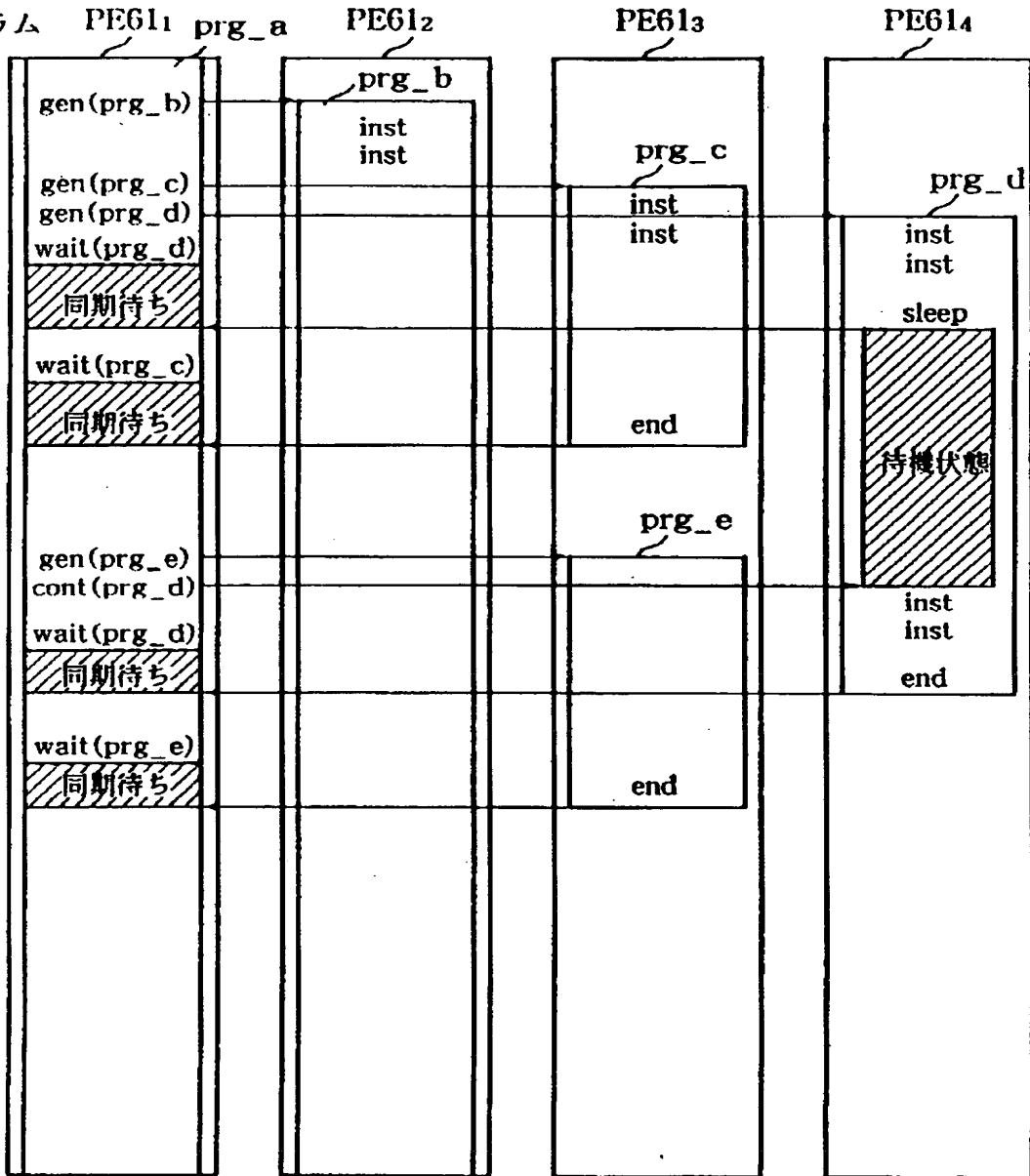
17…共有バス、31…プロセッサコア、32…ローカルメモリ、51…マルチプロセッサ、61₁～61₄…プロセッサエレメント、65…共有メモリ、66…アービタ

【書類名】 図面

【図 1】

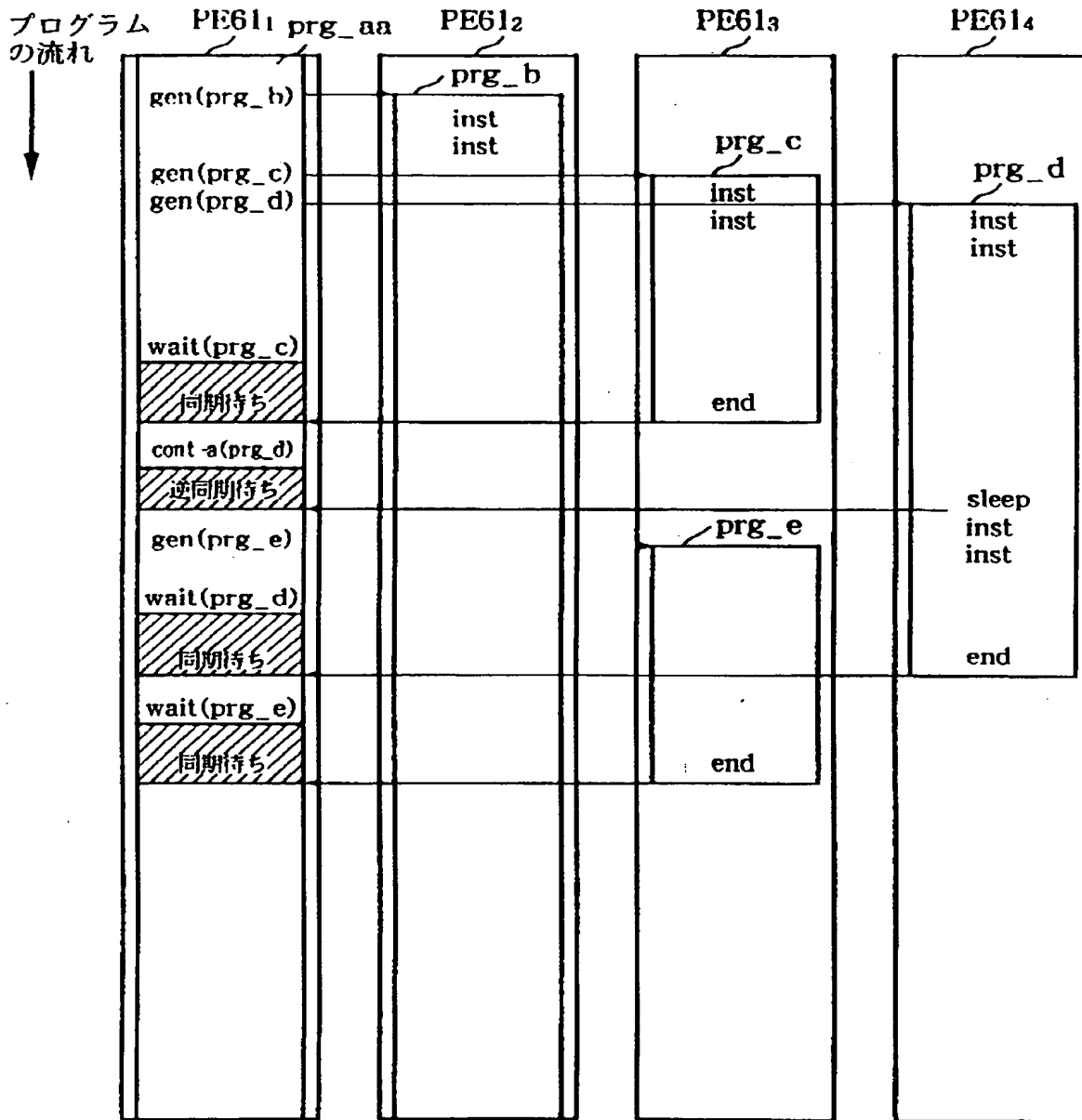


【図 2】
プログラムの
流れ



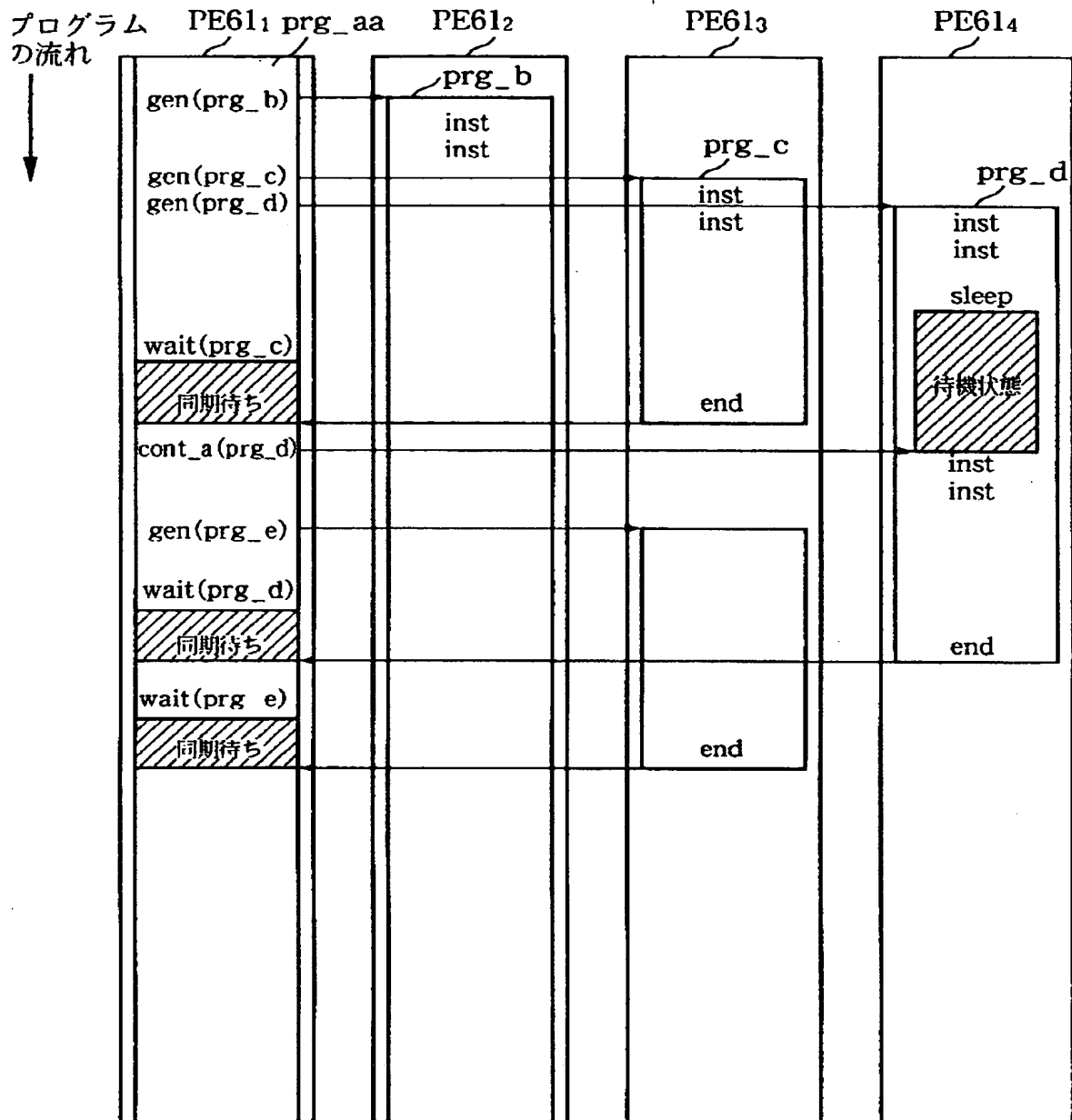
「sleep」: 待機命令
 「cont」: 待機解除命令
 「wait」: 同期待ち命令
 「gen」: プログラム実行指示命令
 「end」: プログラム終了命令

【図 3】

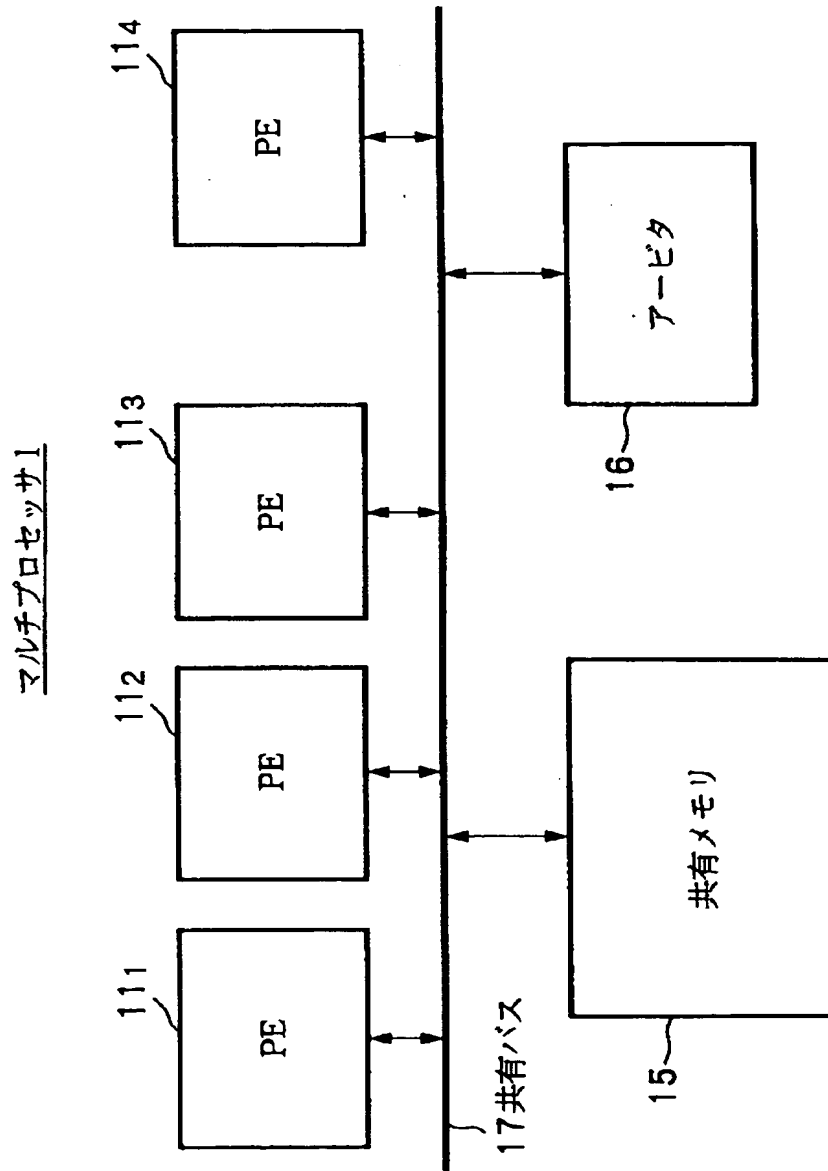


「cont_a」: 待機解除命令

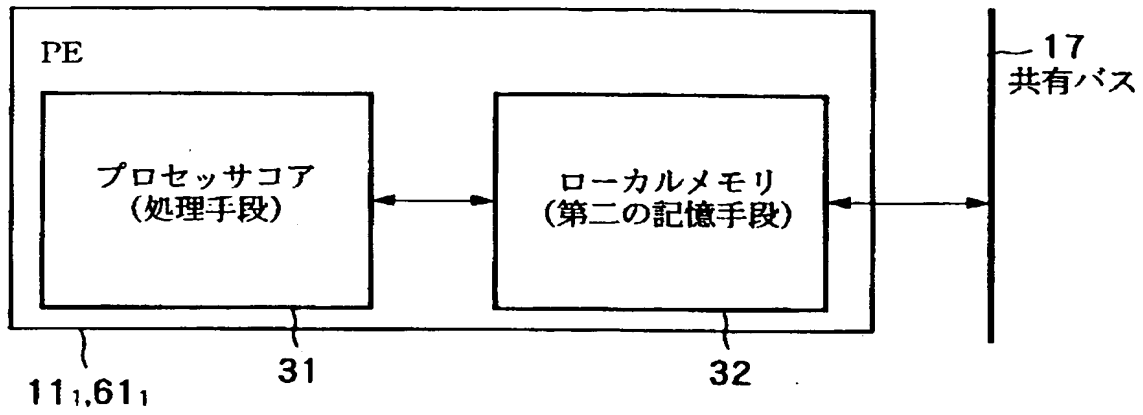
【図4】



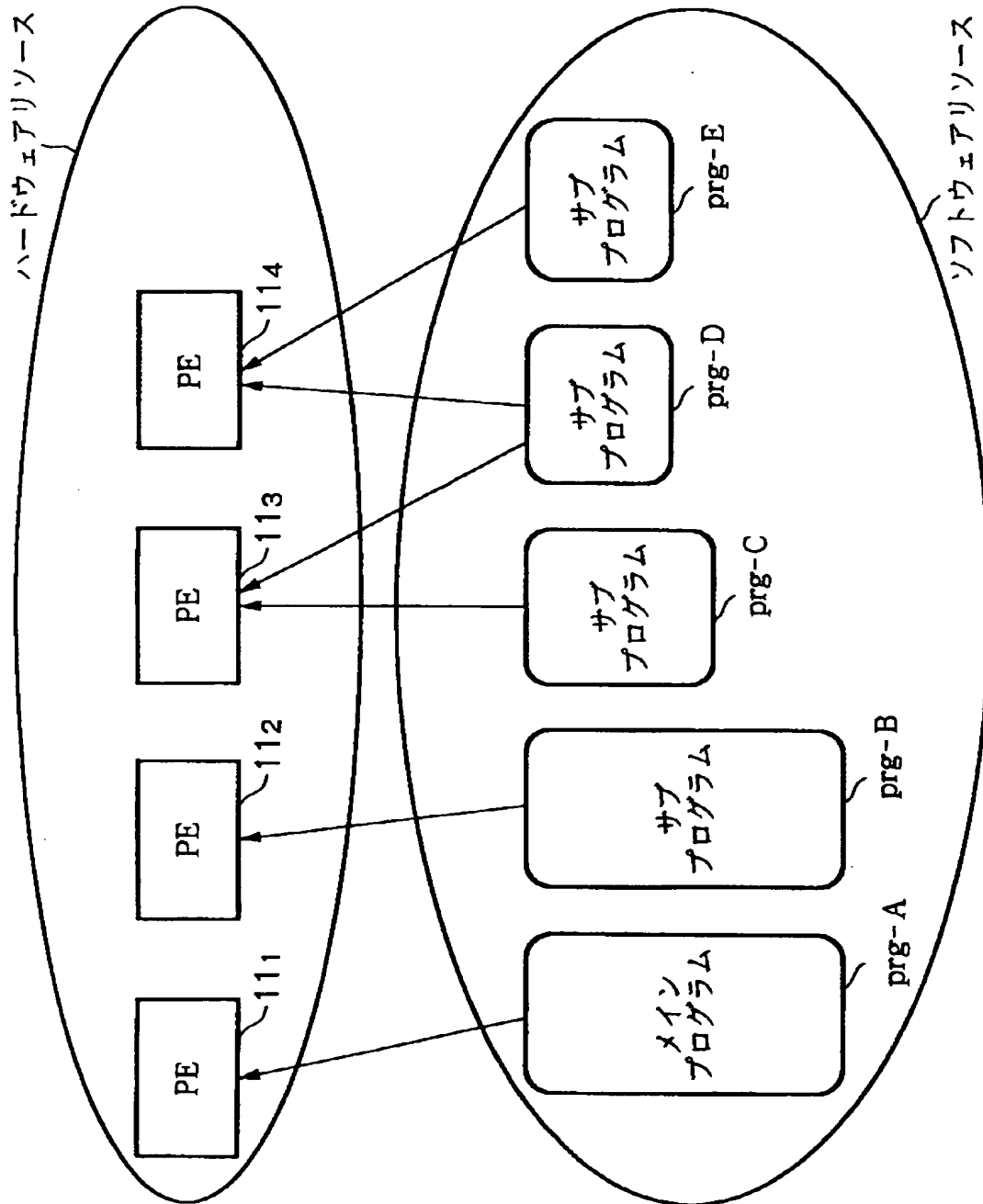
【図 5】



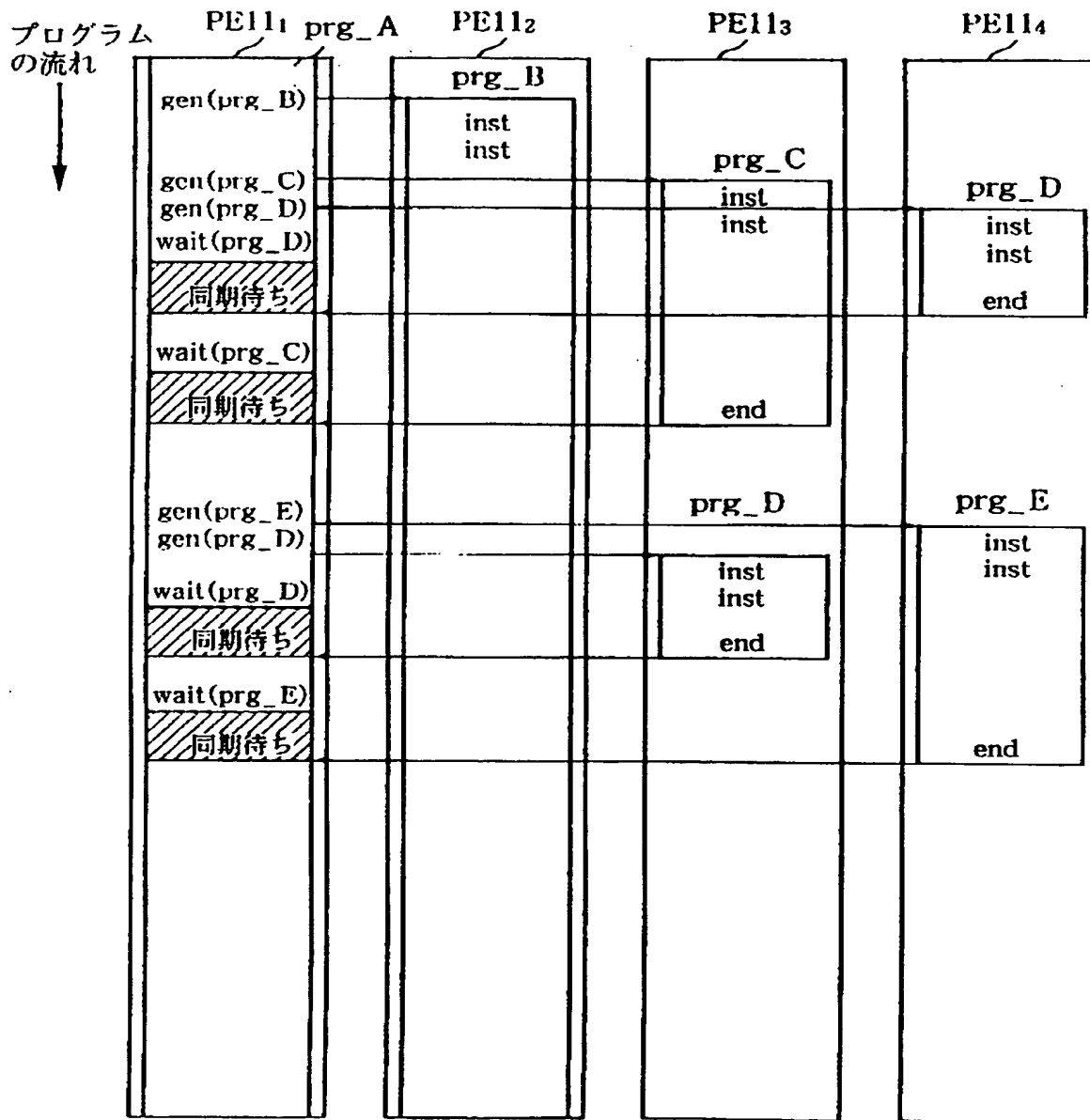
【図 6】



【図 7】



【図 8】



「end」: プログラム終了命令

【書類名】 要約書

【要約】

【課題】 並列に実行されるプログラム相互間で、多様な形態の同期をとることができる並列処理装置を提供する。

【解決手段】 プロセッサエレメント 61₄ は、ユーザプログラム Prg__d において待機命令「sleep」を実行したときに処理を中断して待機状態になり、プロセッサエレメント 61₁ による待機解除命令「cont (Prg__d)」の実行に基づいて前記待機状態を解除して処理を再開し、プロセッサエレメント 61₁ は、待機解除命令「cont (Prg__d)」を実行後、処理を中断することなく次の命令を実行する

【選択図】 図 2

【書類名】 職権訂正データ
【訂正書類】 特許願

<認定情報・付加情報>

【特許出願人】
【識別番号】 000002185
【住所又は居所】 東京都品川区北品川 6 丁目 7 番 3 5 号
【氏名又は名称】 ソニー株式会社
【代理人】 申請人
【識別番号】 100094053
【住所又は居所】 東京都台東区柳橋 2 丁目 4 番 2 号 創進国際特許事務所
【氏名又は名称】 佐藤 隆久

出 願 人 履 歴 情 報

識別番号 [000002185]

1. 変更年月日 1990年 8月30日

[変更理由] 新規登録

住 所 東京都品川区北品川6丁目7番35号

氏 名 ソニー株式会社